

Prof. Lăcrămioara Tufescu

Informatică - Teorie 3

Algoritmi. Structura liniara. Structura decizionala simpla

	Obiective - la sfârșitul lecției vei fi capabil să:	Realizat Da/Nu	Grad de realizare
1	să identifice proprietatile algoritmilor		
2	să identifice etapele de rezolvare a unei probleme		
3	să cunoști operațiile ce se pot efectua în cadrul unui algoritm		
4	să cunoști sintaxa structurii liniare		
5	să cunoști sintaxa structurii decizionale simple		
6	să identifice tipurile de date		
7	să cunoști diferența dintre variabile și constante		
8	să scrii algoritmi cu structuri liniare și decizionale simple		

1 DEFINITIE

Algoritmul este o noțiune matematică foarte veche. Cuvântul *algorithm* este de origine arabă. El derivă din numele matematicianului “Abu Ja’far Mohammed ibn Mûsâ al Horezmi” care a scris o carte celebră intitulată “Kitab al jabr w’al - muquabala”. Din titlul acestei cărți provine cuvântul *algebră*.

În Evul Mediu, noțiunea de algoritm era utilizată de matematicieni pentru a descrie o regulă pe baza căreia se efectuau calcule matematice. Odată cu dezvoltarea calculatoarelor cuvântul *algorithm* a dobândit o semnificație mult mai largă iar gândirea algoritmică este fundamentală pentru rezolvarea problemelor din domenii diverse.

Definiție: Un **algoritm** reprezintă o metodă de rezolvare a unei probleme.

Mărimile cu care lucrează un algoritm sunt:

- **date de intrare** (sunt datele sau valorile cunoscute din textul problemei)
- **date de ieșire** (sunt valorile care trebuie obținute la sfârșitul rezolvării problemei)

A rezolva o problemă înseamnă a obține, din datele de intrare, datele de ieșire, adică soluția problemei.

Algoritmul descrie pas cu pas, în ordinea corectă, succesiunea de operații care arată modul prin care datele de ieșire se obțin din datele de intrare.

Ex. 1: Să facem un ceai fierbinte având la dispoziție un aragaz, un ibric, o cană, linguriță, apă, un pliculeț de ceai și zahăr

Date de intrare: aragaz, ibric, cană, apă, pliculeț de ceai, zahăr

Date de ieșire: cana cu ceai fierbinte

Pas 1. Aprinde aragazul

Pas 2. Pune apa în ibric

Pas 3. Pune ibricul pe foc

Pas 4. Încălzește apa

Pas 5. Pune pliculețul de ceai în cană

Pas 6. Toarnă apa fierbinte peste pliculeț

Pas 7. Pune zahăr în cană

Pas 8. Amestecă cu lingurița până se topește zahărul

Pas 9. Servește ceaiul

Putem inversa pasul 2 cu pasul 7? Dar pasul 5 cu pasul 7?

Ex. 2: Andrei are **A** lei, suficienți pentru a-și cumpăra de la automatul din Mall, o înghețată care costă **P** lei. Andrei introduce banii și apasă butonul pentru înghețată. Câți bani trebuie să-i returneze automatul după ce l-a servit? Scrieți pașii algoritmului care rezolvă problema.

Date de intrare: A și P

Date de ieșire: rest (suma rămasă)

pas1. citește A, P

pas2. $\text{rest} \leftarrow A - P$

pas3. scrie "Restul=", rest

1. Se poate aplica acest algoritm și pentru Bianca?

2. Cum s-ar modifica pașii algoritmului dacă nu i-ar ajunge banii pentru înghețata dorită?

Dar dacă s-a terminat înghețata?

2 PROPRIETĂȚILE ALGORITMILOR

- **claritate** - fiecare operație care se execută la un moment dat trebuie să fie unic determinată, definită și realizabilă (calculatorul nu face față confuziilor);
Ex: nu poate da rest înainte de a primi banii
- **generalitate** - un algoritm trebuie să poată furniza date de ieșire pentru o anumită problemă, pentru mai multe seturi de date de intrare ()
Ex: același automat poate servi unul câte unul, mai mulți copii, care introduc sume diferite de bani și pentru care calculează restul în același mod
- **finitudine** – soluția problemei se poate obține după un număr finit de pași Ex: după ce copilul a luat înghețata, automatul îi dă restul, dacă a introdus mai mulți bani decât costul unei înghețate apoi poate prelua cererea unui nou client
- **corectitudine** – soluția problemei este corectă
Ex: înghețata dată oricărui client este la fel iar restul returnat este corect calculat
- **eficiența** – soluția problemei este obținută în cel mai scurt timp
Ex: IMEDIAT după ce copilul a luat înghețata (NU după câteva zile :-)), automatul îi dă restul, dacă a introdus mai mulți bani decât costul unei înghețate

OBS: Corectitudinea NU este același lucru cu eficiența!

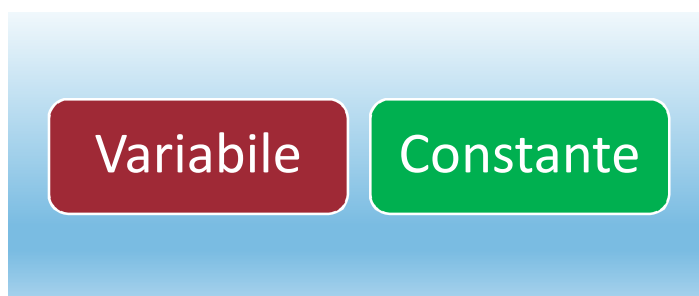
- 📌 Ce s-ar întâmpla dacă la căutarea unei informații pe net, răspunsurile primite ar fi corecte dar ar apărea pe ecran după 3 zile??
- 📌 Dacă sunt șofer în trafic, nu știu drumul și folosesc GPS-ul pentru a ajunge la destinație, ce s-ar întâmpla dacă GPS-ul m-ar anunța că trebuie să fac o curbă la dreapta după câteva ore după ce am trecut de locul respectiv??

2.1 ETAPELE REZOLVĂRII UNEI PROBLEME SUNT:

1. **Analiza problemei**, se referă la:
 - ✓ identificarea datelor de intrare și a datelor de ieșire
 - ✓ stabilirea operațiilor care trebuie efectuate asupra datelor de intrare, pentru a obține datele de ieșire
2. **Scrierea algoritmului de rezolvare a problemei**, se referă la descrierea operațiilor pe pași (ca și în ex.1 și 2).
3. **Implementarea algoritmului într-un limbaj de programare**, se referă la scrierea programului C++ care va rezolva problema.
4. **Verificarea corectitudinii algoritmului** – acest lucru înseamnă executarea programului pentru diferite seturi de date de intrare.
5. **Analiza complexității algoritmului** – eficiența unui algoritm are în vedere atât spațiul de memorare necesar a fi ocupat pe durata execuției algoritmului, cât și numărul de operații executate de acesta.

3 OBIECTELE CU CARE LUCREAZĂ ALGORITMI SUNT: DATE ȘI EXPRESII.

3.1 DATELE POT FI:



a) variabile

O variabilă are alocată o zonă de memorie, are un nume, un tip și o valoare atașată.

De exemplu, **A**, **P**, **rest** sunt variabilele ce intervin în ex 2.

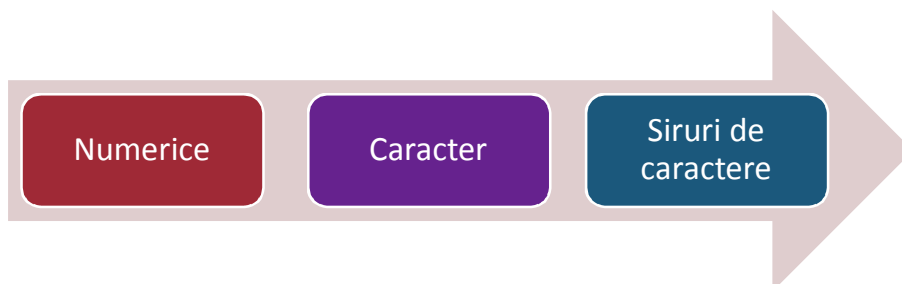
Numele unei variabile trebuie să înceapă cu o literă și poate conține doar litere cifre și simbolul '_'.

Exemplu:

- ✓ a, maxim, nota_1, a2011 – CORECTE;
- ✓ a 5-a B, \$, 12A - GREȘITE

b) **constante** – sunt date care nu-și schimbă valoarea pe parcursul execuției programului.

După **tipul** lor, datele pot fi:



3.2 EXPRESIILE

O expresie este formată dintr-o succesiune de **operanți** intercalați de **operatori** (simboluri pentru operații).

Ex:

$$25 - 7 + 32$$

operanți: 25, 7 și 32
operatori: - și +
valoarea expresiei: 50

$$15 / 2 + 15 \% 2$$

operanți: 25 și 2
operatori: / (câtul împărțirii a două numere naturale),
% (restul împărțirii a două numere naturale)
+ (adunarea a două numere)
valoarea expresiei: $15 / 2 = 7$ (câtul împărțirii lui 15 la 2)
 $15 \% 2 = 1$ (restul împărțirii lui 15 la 2)
 $7 + 1 = 8$

3.3 EVALUAREA UNEI EXPRESII = ÎNLOCUIREA VARIABILELOR CU VALORILE LOR ȘI CALCULAREA VALORII OBTINUTE PRIN APLICAREA CORECTĂ A OPERATORILOR DIN EXPRESIE

Ce valoare are expresia următoare: $E = (a + b * 3) * a - 5 * b$ pentru $a = 4$ și $b = 1$?

3.4 OPERAȚII EFECTUATE ÎN CADRUL ALGORITMULUI

În cadrul algoritmului, cu datele putem efectua:

- **operația de citire** – realizează introducerea de date în memoria calculatorului;
- **operația de scriere** – realizează extragerea rezultatelor din memoria calculatorului și scrierea lor (pe ecran sau într-un fișier pe Hard disk);
- **operația de atribuire** – permite stabilirea unor valori inițiale și efectuarea de calcule;
- **operația de decizie** – în funcție de îndeplinirea sau neîndeplinirea unei condiții, algoritmul se ramifică.

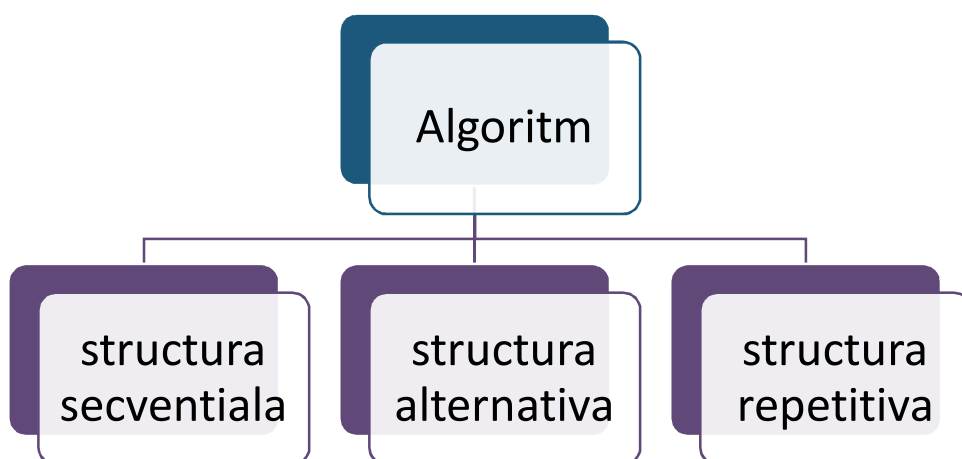
4 REPREZENTAREA ALGORITMILOR ÎN PSEUDOCOD

Limbajul pseudocod reprezintă o modalitate foarte utilizată de reprezentare a algoritmilor.

Acesta folosește o serie de **cuvinte-cheie** în descrierea algoritmilor. **Cuvintele-cheie** sunt cuvinte cu o semnificații prestabilite; ele participă în descrierea operațiilor ce formează algoritmul.

În ex.1 cuvintele-cheie utilizate au fost: **citește, scrie**.

Orice algoritm poate fi reprezentat folosind următoarele tipuri de structuri de control.



➤ **Structura secvențială**

operație 1
operație 2
.....
operație n

unde operație 1, ..., operație n pot fi operații de citire, scriere, atribuire

atribuire: **variabila** ← expresie

spunem că "**variabila** PRIMEȘTE valoarea **expresie**" (se evaluează expresia din dreapta iar valoarea obținută se depune în variabila din stânga)

➤ **Structura alternativă simplă**

dacă expresie
 atunci execută instrucțiune_1
altfel execută instrucțiune_2

➤ **Structura repetitivă** – va fi detaliată în fișa următoare

5 APLICAȚII

Scrieți algoritmi și programe C++ pentru rezolvarea următoarelor probleme:

5.1 CUMPĂRĂTURI

Un client trebuie să plătească cumpărăturile făcute la supermarket. Casiera constată că a cumpărat 2 tipuri de produse dar a luat din fiecare tip mai multe bucăți. Casa de marcat va „afla” prețul unui tip de produs după ce casiera scanează o bucată cu ajutorul cititorului de coduri de bare apoi va introduce de la tastatura casei de marcat numărul de bucăți la fel. Va proceda în același mod și pentru celălalt produs. La sfârșit, pe ecran trebuie să apară valoarea totală de plată.



Scrieți un algoritm / program C++ care citește prețul unei bucăți din fiecare tip și numărul de bucăți la fel și afișează suma totală de plată.

Algoritm

date de intrare: pret1, nrbuc1, pret2, nrbuc2

date de ieșire: total

citeste pret1, nrbuc1, pret2, nrbuc2

total \leftarrow pret1 * nrbuc1 + pret2 * nrbuc2

scrie "total de plată = ", total, " lei. "

Sfarsit

Programul C++

```
#include <iostream>
using namespace std;
int main()
{
    int pret1, nrbuc1, pret2, nrbuc2, total;
    cout << "cat costa o bucata din produsul de tip 1? ";
    cin >> pret1;
    cout << "cate produse de tip 1 a cumparat? ";
    cin >> nrbuc1;
    cout << "cat costa o bucata din produsul de tip 2? ";
    cin >> pret2;
    cout << "cate produse de tip 2 a cumparat? ";
    cin >> nrbuc2;
    total = pret1 * nrbuc1 + pret2 * nrbuc2;
    cout << "total de plata = " << total;
    return 0;
}
```

Obs:

cin – permite citirea datelor de intrare de la tastatură

cout – permite scrierea datelor de ieșire pe monitor

Mesajul cuprins între " " (ghilimele) se va afișa pe ecran așa cum a fost scris

5.2 AUTOMATUL DE ÎNGHEȚATĂ

Un automat de înghețată oferă un singur sortiment de înghețată care costă **P** lei. Orice client va introduce o bancnotă de valoare **B** și apăsa butonul de comandă. Scrieți un algoritm / program care analizează cererea clientului și afișează pe ecranul automatului un mesaj corespunzător de tipul: „*Vă multumim pentru alegerea făcută! Poftă bună! Ridicați restul de ... lei.*” sau „*Comanda nu poate fi onorată. Mai introduceți ... lei.*” Considerăm că automatul de înghețată este mereu plin și poate da rest la orice tip de bancnotă.



Algoritm

date de intrare: P, B

date de ieșire: mesaj

citeste P, B

daca $P \leq B$

atunci scrie "Pofta buna! Ridicati restul de ", B-P, " lei."

altfel scrie "Plată insuficienta. Mai introduceți ", P-B, " lei"

sfarsit

Programul C++

```
#include <iostream>
using namespace std;
int main()
{
    int B,P;
    cout << "cat costa inghetata? ";
    cin >> P;
    cout << "valoarea bancnotei introduse? ";
    cin >> B;
    if (P <= B) cout << "Pofta buna! Ridicati restul de " << B-P << " lei." << endl;
    else cout << "Plata insuficienta. Mai introduceți " << P-B << " lei." << endl;
    return 0;
}
```

5.3 TEMA: GREUTATEA IDEALĂ

Se știe că greutatea ideală se calculează după formula de mai jos

$$\text{Greutatea_ideala} = (\text{inaltimea} - 100) + (\text{varsta} / 10) * 90/100$$

unde înălțimea este dată în centimetri iar vârsta în ani împliniți. Scrieți un algoritm și un program C++ care să calculeze și să afișeze greutatea ideală pentru o persoană pentru care se dă înălțimea H și vârsta V.

